IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

GEMALTO S.A.,	§	
	§	
Plaintiff,	§	
v.	§	Civil Action No. 6:10-CV-561-LED
	§	
HTC CORPORATION, HTC AMERICA, INC., EXEDEA, INC., SAMSUNG ELECTRONICS CO., LTD., SAMSUNG TELECOMMUNICATIONS AMERICA LLC, MOTOROLA MOBILITY, INC., and GOOGLE INC.,	§	JURY TRIAL DEMANDED
	§	
Defendants.	§	

GEMALTO'S OPENING CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

TABLE OF AUTHORITIES

TABLE OF AUTHORITIES (Cont.)

Index of Exhibits

Ex. A	United States Patent Number 6,308,317
Ex. B	*JavaCard Technology Grows Up Smart* by Janice Heiss and John Papageorge
Ex. C	Email dated October 31, 1996 From Bertrand du Castel
Ex. D	October 15-17 Best Innovation Award
Ex. E	Press release dated November 9, 2007
Ex. F	www.javacardforum.org welcome page
Ex. G	*Personal History of the Javacard* by Bertrand du Castel
Ex. H	*The Java Language Environment*, A White Paper, May 1996 by James Gosling
Ex. I	Preliminary Amendment Dated July 25, 2011
Ex. J	May 27, 2004 Reply Brief submitted during the prosecution of the ’485 patent
Ex. K	Amendment and Response Dated August 28, 2000
Ex. L	United States Patent Number 7,117,485
Ex. M	*The Microcontroller Idea Book* by Jan Axelson
Ex. N	United States Patent Number 5,796,312
Ex. O	United States Patent Number 5,809,544
Ex. P	United States Patent Number 6,260,101
Ex. Q	United States Provisional Application Number 60/029,057
Ex. R	United States Patent Number 7,818,727
Ex. S	United States Patent Number 6,687,898
Ex. T	United States Patent Number 6,990,662
Ex. U	United States Patent Number 7,801,068
Ex. V	Petition for Correction Dated July 30, 2010

Ex. W	Petition for Correction Dated February 2, 2011
Ex. X	Reply and Amendment Dated February 21, 2008
Ex. Y	Office Action Dated December 21, 2007
Ex. Z	United States Patent Number 6,256,663

I. Introduction

Pursuant to P.R.4-5(a), Gemalto S.A. ("Gemalto") respectfully submits this opening brief on the proper construction of disputed claim terms of the "Gemalto Patents".[1] All of the Gemalto Patents share a common specification and claim priority back to an October, 25, 1996 Provisional Application entitled "Implementation of the Java Programming Language for Microcontrollers." While Gemalto currently asserts 100 of the 158 claims in the Gemalto Patents,[2] there are presently only 24 disputed terms and those terms are used consistently throughout those claims. All 24 of those disputed terms are present in a subset of 8 "representative claims," reproduced in Appendix 1 for the Court's convenience.

II. Overview of Technology and Patents

The technology claimed in the Gemalto Patents was invented in the mid-1990s at Gemalto's corporate predecessor, Schlumberger, at its Austin smartcard facility. While the technology is not limited to smartcards, but extend to any microcontroller application, it is instructive to understand smartcards and the Java programming language to appreciate the invention and what the inventors accomplished.

a. Smartcards

A smartcard is a secure, robust, tamper-resistant and portable device for storing data. It typically includes a microprocessor and memory for storing and executing a small piece of computer code, all mounted on a plastic card. At the time, smartcards used an application-specific microprocessor called a "microcontroller" that typically had "a small RAM of 0.1 to

[1] The Gemalto Patents are U.S. Patent Nos. 6,308,317 ("the '317 Patent"), 7,117,485 ("the '485 Patent") and 7,818,727 ("the '727 Patent").

[2] Gemalto is committed to reducing the number of asserted claims once it has completed its discovery, Defendants' have reduced their "hide the ball" invalidity contentions (comprising hundreds of references and combinations) to what reasonably can be used at trial, and received the Court's claim construction, as set forth in Gemalto's opposition to Defendants' motion to limit the number of claims filed concurrently herewith. (*See* D. I. 162.)

2.0K, 2K to 8K of EEPROM, and 8K-to56K of ROM." (Ex. A, 2:33-35.)[3] In that sense, the smartcard's computing capabilities made it the smallest and most personal of computers. Smartcards were originally used as debit cards and became prominent in the financial world, especially in Europe. They were later adopted by telecommunication providers as Subscriber Identity Modules (SIM) cards for use in authenticating GSM phones to the network.

Prior to the Gemalto inventions, applications for smartcards were typically programmed directly in low level computer programming languages (e.g., assembly language) in order to maximize the computing resources available on the card. These low level programming languages were usually proprietary and closely tied to the specific hardware of the smartcard. If the hardware changed, the programming language changed. While this allowed for efficient use of the limited computing resources on a smartcard, it made the development more difficult and limited the pool of potential application developers. In order to solve this problem, researchers looked for a way to program smartcards in a high level programming language.

b. Java—"Write Once, Run Anywhere"

As smartcard developers were looking for ways to enhance the programming environment, there was a revolution occurring in the PC world. In 1995, Sun Microsystems released Java, a high level programming language that was immediately and widely heralded for its "write once, run anywhere" philosophy. This meant that an application developer could write in <u>one</u> language and compile the language to class files containing interpretable byte codes to run on <u>any</u> platform through software called a Java Virtual Machine (VM) that was designed to speak the disparate proprietary languages of the different computing platforms, *e.g.*, PC's, Sun Workstations, Apple computers. In a Java environment, a developer only needs to create one set of byte codes and test them for one VM—this same set of byte codes will then execute on any

[3] All citations to patents are in a "column:line" format and refer to the '317 Patent (Ex. A).

VM running on any computing platform without further re-compiling or testing. Because of this "write once, run anywhere" characteristic, Java quickly became the high-level language of choice for application programmers and would revolutionize the World Wide Web.

c. JavaCard's Technical Challenges—"Like Playing Golf in a Phone Booth"

Java, however, was designed to operate on general purpose computing platforms, *i.e.*, on "microprocessor-based computers, with access to relatively large amounts of memory" such as found "in desktop and personal computers." (Ex. A, 1:56-58.) Now, in addition to the operating system and the application code, the smartcard would have to provide code for the "Java Virtual Machine," i.e., "JVM," to interpret the Java byte codes into native code (the proprietary language) that could be executed by the actual processor of the computing platform. Sun's CEO, Scott McNealy, later characterized the problem as follows: "fitting Java technology inside smartcards was like playing golf in a phone booth." (Ex. B, "JavaCard Technology Grows Up Smart" by Janice Heiss and John Papageorge, p.1) To tackle this seemingly insurmountable obstacle, Schlumberger assembled a team of world-class computer scientists including Dr. Tim Wilkinson, Dr. Scott Guthery, Dr. Ksheerabdhi Krishna and Mike Montgomery. Their solution included, among other things, innovative techniques for converting the Java byte codes into byte codes that minimize the computing resources consumed by both the application and the JVM so that it works within the resource constraints of the smartcard. (Ex A, 8:40-11:48.)

d. JavaCard's Commercial Success—Billions Sold

Schlumberger's innovation was met with immediate industry recognition and acclaim. Schlumberger announced the first-ever Java-based smartcard prototype at the world-wide Cartes 96 smartcard convention in October 1996. (Ex. C, 10/31/96 du Castel e-mail.) The following year at Cartes 97, Schlumberger received the "Best Innovation" award from its industry peers for its invention. (Ex. D, Cartes 97 Award.) The industry quickly followed Schlumberger's lead by establishing the JavaCard Forum to standardize this technology and regain competitive footing in this new area. (Ex. E, JavaCard Forum Press Release; Ex. F, JavaCard Website Introduction) By 2007, over 5 billion JavaCards had been sold for use in credit cards, cell phones and identification cards across the globe. (Ex. G, "Personal History of the Javacard.")

Schlumberger eventually spun off its smartcard division into Axalto, which subsequently merged with Gemplus, another smartcard supplier, in 2008 to form Gemalto. Gemalto, by itself and through its predecessors, has actively licensed its JavaCard technology for smartcards, including to the creator of the Java programming language, Sun Microsystems, now part of Oracle, who unsuccessfully sought to invalidate the '317 Patent in a reexamination filed in 2006.

III. Argument : The Court Should Adopt Gemalto's Proposed Constructions

All of the asserted claims are directed to a two-step process for transforming an application from a high level programming language into a form that can be executed on a virtual machine by the associated processor. In the first step, an application written in a "high-level programming language" is "compiled" into a "compiled form" or "intermediate form." (*See generally,* Appendix 1.) This "compiled form" is then transformed into a "converted form" using the algorithms shown in Figures 5 & 6 (Ex. A, Figs. 5 & 6; 8:40-11:48) that can be executed on an "integrated circuit card," "programmable device" or even within the "resource constraints" of a "microcontroller." (S*ee generally,* Appendix 1.)

a. Compiling Related Terms

1) High level language / High level programming language

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
high level language / high level programming language	programming language that must be compiled and interpreted before it can be executed by a computer	programming language that must be compiled or interpreted before it can be executed by a computer

Gemalto's proposed construction is consistent with the way these terms are repeatedly used throughout the claims and the specification of the Gemalto Patents. *Bell Atl. Network*

Servs. v. Covad Communs. Group, 262 F.3d 1258, 1271 (Fed. Cir. 2001) ("[W]hen a patentee uses a claim term throughout the entire patent specification, in a manner consistent with only a single meaning, he has defined that term 'by implication.'"). The context provided in the claims and specification implicitly defines a high level programming language as Java and any other programming language that must be <u>compiled</u> <u>and</u> interpreted before it can be executed by a computer, not simply compiled <u>or</u> interpreted as defendants propose. (*See Ex. A*, 1: 24-34 and 18:47-64 ("When a Java application is written, it is *complied* into 'Class' files containing byte codes that are instructions for a hypothetical computer called a Java Virtual Machine. An implementation of this virtual machine is written for each [computing] platform that is supported …The Java virtual machine for the selected platform is run, and *interprets* the byte codes in the class file, thus effectively running the Java application. The term applications includes any program, such as Java applications … and other non-Java programs that can result in class files as described below. Class files may have a source other than Java program files. Several programming languages other than Java also have compilers or assemblers for generating class files from their respective source files…. Regardless of the source of the class files, the above description applies to languages other than Java to generate codes to be interpreted.").)

2) Compiling / Compiled Form

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
compiling	transforming a program written in a high-level programming into class files containing byte codes	transforming a program written in a high-level programming language from source code to object code or byte code
compiled form	class files containing byte codes	No construction necessary

The primary difference between the parties' constructions of the terms "compiling" and "compiled form," is in Defendants' attempt to broaden the scope of these terms from transforming an application into "class files containing byte codes" (as repeatedly and consistently set forth in the specification and claims) to also include "object code." Not a single claim recites the term "object code" as defendants propose in their definition, nor does the term "object code" appear anywhere in the specification—and for good reason: it cannot be disputed by defendants that compilers that compile an application programming language into object code are transforming the program into codes that are directly executable by a specific computing platform and thus are not interpretable by a virtual machine. Thus, such "object code" compilers fall outside the scope of the invention described and claimed in the Gemalto Patents.

Unlike "object code" which is only executable on a specific-computing platform, "class files containing byte codes" are executable on a whole host of computing platforms because such interpretable codes are compiled for a virtual machine, not a specific computing platform. That virtual machine is designed to interpret such byte codes into the machine language of many disparate computing platforms. (*See, e.g.,* Ex. A, 1:24-34 and 18:47-64 ("When a Java application is written, it is compiled into *'Class' files containing byte codes* that are instructions for a hypothetical computer called a Java Virtual Machine. An implementation of this virtual machine is written for each [computing] platform that is supported….Class files may have a source other than Java program files. Several programming languages other than Java also have compliers or assemblers for generating class files from their respective source files…. Regardless of the source of the class files, the above description applies to languages other than Java to generate codes to be interpreted.").) All embodiments described in the specification compile an application into interpretable byte codes, not Defendants' proposed object codes (the directly executable zeros and ones of a specific computing platform).[4]

[4] The specification of the Gemalto Patents also incorporates by reference certain Java programming language reference materials (see Ex. A, 1:35-47) that further emphasize the distinction between computing platform-specific object codes and the computing platform-independent class files containing byte codes:"[t]he Java compiler doesn't generate 'machine code' in the sense of native hardware instructions [zeros and ones]—rather, it generates bytecodes: a high-level, machine-*independent* code for a hypothetical machine that is implemented by the Java interpreter and run-time system." (Ex. H, *The Java Language Environment,* A White Paper, May 1996 by James Gosling, p. 51.)

Moreover, during prosecution of a continuation of the ’727 Patent (i.e., the ‘949 application), the patentee distinguished “object code” compilers from the claimed invention. (*See*, *e.g.*, Ex. I, 7/25/11 Amendment, at 29-30;(“while [the Chaitin reference] discusses ‘converting’ files, the conversion therein is not the same as the conversion in the present application in that conversion [there] refers to conversion between different file formats in order to make an object [code] module for a specific hardware platform *directly executable* as opposed to conversion from an *interpretable compiled code* … as is the case in the present claims.”).) This statement further highlights that the compiled code of the present invention does not—nor was it ever intended to—include object code.

In contrast, Defendants merely rely upon dictionary definitions for their proposed constructions that are inconsistent with the intrinsic record discussed above and basic claim construction principles that put the intrinsic record first. The only other support mentioned by Defendants for including object code is a remark in the prosecution history taken out of context.[5]

3) Byte Codes and Virtual Machines

Term	**Gemalto’s Proposed Construction**	**Defendants’ Proposed Construction**
byte codes / byte code format	program instructions interpretable by a virtual machine	No construction necessary
virtual machine/ intermediate code virtual machine	program used to interpret byte codes	No construction necessary

These limitations require construction to assist the trier of fact and avoid confusion. Absent construction, Defendants will attempt to equate object codes, which are not interpretable by a virtual machine (and thus outside the scope of the claimed invention), with byte codes

[5] In a May 27, 2004 Reply Brief submitted during the prosecution of the ’485 patent, the patentee distinguished the Renner reference because it did not show any form of compiling. (*See* Ex. J, ’485 File History (“compiling is a term of art of Computer Science meaning ‘To transform a program written in a high-level programming language from source code into object code. Just because computers [in Renner’s system] operate on ones and zeros does [not, sic] mean that all data in the computer has gone through a compilation.’”).)

which are interpretable by a virtual machine. They will do so in an effort to improperly expand the scope of the claims in support of their validity challenge. In addition, while these terms are well known to a person of ordinary skill in the art and play an integral role in the asserted claims, they also share common terminology with other terms understood by a lay person, *e.g.*, "byte codes" versus "bytes," which may further lead to jury confusion.

Gemalto's constructions for "byte code" or "byte code format" and "virtual machine" are derived from the specification which teaches that byte codes "are instructions for a hypothetical computer called a Java Virtual Machine." (Ex. A, 1:26-27.) However, because *non-Java* high level programming languages are taught by the specifications of the Gemalto patents, the proposed construction is not limited to instructions for only a Java Virtual Machine. (*Id.*, 1: 24-34 and 18:47-64) In order to account for all embodiments presented in the Gemalto patents, the proper construction for bytes codes is "program instructions interpretable by a *virtual machine*" and for virtual machine the proper construction is a "computer program used to interpret byte codes." (*Id.*, 1:33-35 ("The Java virtual machine … interprets the byte codes in the class file, thus effectively running [executing] the Java application."))

4) Class File Format

Term	**Gemalto's Proposed Construction**	**Defendants' Proposed Construction**
class file format	a file format containing byte codes that are instructions for a virtual machine	a file format containing byte codes that are instructions for a hypothetical computer
application having a class file format	a program whose compiled form is in a class file format	No construction necessary

The parties agree that "class file format" includes "a file format containing byte codes that are instructions," with Gemalto proposing that these instructions are for a "virtual machine" while Defendants suggest a "hypothetical computer." The specification uses both terms interchangeably. However, Gemalto's use of "virtual machine" seeks to maintain consistent use of terminology among the proposed constructions and claims to avoid jury confusion. The term virtual machine is used repeatedly and consistently throughout the claims and specification of the Gemalto Patents, and in other proposed claim constructions. In contrast, the term "hypothetical computer" is not used at all in the claims and is used only once in the specification of the Gemalto Patents (see below quote).

Gemalto's construction for the term "application having a class file format" is "a program whose compiled form is in a class file format." Gemalto's construction is consistent with the specification and claims of the Gemalto Patents which teach that such an application is a program written in a high level programming language that—when compiled—is compiled into "class files containing byte codes." (Ex. A, 1:24-27, 18:42-46 ("When a Java application is written, it is compiled into *"Class" files containing byte codes that are instructions for a hypothetical computer called a Java Virtual Machine*…. The term application includes any program, such as Java applications, Java applets, Java aglets, Java servlets, Java commlets, Java components, *and other non-Java programs that can result in class files*."); and 8:24-36 ("the [Java] application class files 24 [in FIG.2] follow the standard *class file format* documented in Chapter 4 of the Java virtual machine specification …."); *see also* 18:47-64.)

b. Conversion Related Terms

1) Converting / Converted Form

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
converting / converted form	postprocessing the byte codes of the compiled form into a converted form *converted form* byte codes interpretable by a different virtual machine than the compiled form	No construction necessary

The terms "converting" and "converted form" are at the core of the claimed invention and thus require construction. *See O2 Micro Int'l Ltd. v. Beyond Innovation Tech. Co.*, 521 F.3d 1351, 1360 (Fed. Cir. 2008) ("When the parties raise an actual dispute regarding the proper scope of these claims, the court, not the jury, must resolve that dispute."). The "No construction necessary" proposal from defendants is yet again an effort to leave open the possibility that they can later improperly expand the scope of these claim terms (beyond what was intended in the intrinsic record) or otherwise confuse the jury in support of their validity challenge. Gemalto's construction of the term "converting" is supported by the language of the claims, the specification, and the file history, all of which indicate that "converting" is a postprocessing step that occurs after an application written in a high level programming language is compiled into class files containing byte codes (*i.e*., a class file format). For example, the asserted claims make clear that a program written in a high level programming language is first compiled into byte codes and then converted to converted form. (*See*, *e.g.*, Appendix 1, '317 Patent, claim 1.) The specification teaches that the claimed converter "is a class file *postprocessor* that processes a set of class files that are encoded in the standard Java class file format…to produce a Java card class file in a [a different] card class file format." (Ex. A, 8:40-44.) The prosecution history is also consistent: "[a]pplicants respectfully submit that the conversion process [to a converted form] is a *post processing* step." (Ex. K, '317 File History, 8/28/00 Amendment at 15.)

Gemalto's construction for the term "converted form"—"byte codes interpretable by a different virtual machine than the compiled form" is likewise consistent with the intrinsic record. The specification which teaches that the virtual machine for the converted byte codes is different from the virtual machine for the compiled byte codes. For example, Figure 6 of the patents shows that the conversion operation converts byte codes from one form to a second, different form. (Ex. A, Fig. 6 ("Original byte codes" converted to "*Modified byte codes*").) Likewise, the specification describes that "the card class file converter modifies the original byte codes into *a different set of byte codes* supported by the particular Card JVM 16 being used." (*Id.*, 11:4-6). In addition the specification teaches:

- "The first instructions may include byte codes for a first type of virtual machine, and the second instructions may include byte codes for a second type of virtual machine. *The first type is different from the second type*."(*Id.*, 5:52-56.)
- "[T]he card class file converter modifies the original byte codes into a different set of byte codes designed for *a different virtual machine* architecture."(*Id.*, 11:24-26.)

Indeed, because conversion modifies the byte codes for a specific target virtual machine—*i.e.*, for a microcontroller, integrated circuit card, or programmable device—the converted form must be byte codes interpretable by a different virtual machine than the compiled form—*e.g.*, virtual machines for personal computers or workstations.

2) **Attributes**

Term	**Gemalto's Proposed Construction**	**Defendants' Proposed Construction**

attributes	No construction necessary. To the extent that the term is construed, it should be given its plain and ordinary meaning:
	
	information in a class file
		Data structures having the following general format:
		attibute_info {
		u2 attribute_name_index;
		u4 attribute_length;
		u1 info[attribute_length];}

One of the conversion techniques taught by the patent involves removing attributes from the compiled form. (*See, e.g.*, Appendix 1, '317 Patent, claim 66.) Gemalto respectfully submits a plain and ordinary construction for "attributes" ("information in a class file") while Defendants seek to improperly limit the construction to the specific "data structures" described in the Java Specification. The specification makes clear, however, that attributes are information in a class file and that such attributes should not be limited to those described in the Java Specification: "typical application class file includes… various *attribute* information, as detailed in the aforementioned Java Virtual Machine Specification." (Ex. A, 8:62-66.) As discussed at length above, however, the scope of the claimed invention also includes applications written in non-Java high level programming languages which have different attribute information than the Java Specification. (*Id.*, 1: 24-34 and 18:47-64.) Accordingly, the term should not be limited to the precise format described in the patent for the Java programming language, as Defendants propose. *Playtex Prods., Inc. v. Procter & Gamble Co.*, 400 F.3d 901, 908 (Fed. Cir. 2005).

3) **Specific Byte Code / Generic Byte Codes**

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
specific byte codes / generic byte code	*specific byte code* a byte code with a built-in argument or operand *generic byte code* a byte code with a separate, accompanying argument or operand	smaller byte codes / larger bytes codes for the same operation

Another conversion technique presented in the Gemalto patents and claims involves converting "specific byte codes" to "generic byte codes." Figure 7 provides an example with respect to one specific byte code, "ILOAD_0" and one generic byte code "ILOAD." The Gemalto Patents describe this process as follows:

> This generally entails the translation of short single specific byte codes such as ILOAD_0 into their more general versions. For example, ILOAD_0 may be replaced by byte code ILOAD with an argument 0.".

(Ex. A, 10:40-44.) The ILOAD_0 byte code is a specific byte code, whose argument (used interchangeably in the specification with "operand"), '0', is already built in, i.e., it has been predefined with the byte code in the class file. In contrast, ILOAD is a generic byte code that is missing this built-in operand and requires that the operand separately accompany it. This way one generic byte code may be used instead of a series of specific byte codes that include each of the different operands. "This translation is done to reduce the number of byte codes translated by the Card JVM 16, consequently reducing the complexity and code space requirements for the Card JVM 16." (Ex. A, 10:44-47.) Gemalto's construction tracks the language in the specification while Defendants' does not.

c. Apparatus Related Terms

The asserted claims can be grouped into three distinct apparatus: a "microcontroller," an "integrated circuit card," and a "programmable device." Each is a different device with a well-known and accustomed meaning in the art to which the parties largely agree. However, Defendants improperly attempt to load these terms up with additional, unnecessary limitations that are not supported by the intrinsic record and to read a broad subject matter disclaimer into their constructions. Gemalto's plain and ordinary meaning of these terms should be adopted.

1) Microcontroller[6]

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
microcontroller	a device designed specifically for embedded applications that includes a central processing unit, memory and other functional elements on a single semiconductor substrate, or integrated circuit	a single semiconductor substrate or integrated circuit that includes a central processing unit, memory and other functional elements, and that does not require external components to function properly; not a microprocessor

Both parties agree that a "microcontroller" includes a central processing unit, memory and other functional elements on a single semiconductor substrate or integrated circuit. Consistent with the specification and its ordinary meaning, a microcontroller is specifically designed for embedded applications (i.e., specific tasks or purposes). Defendants, however, introduce two negative limitations: "does not require external components to function properly" and "not a microprocessor." Defendants' negative limitations are improper.

The specification states that "a microcontroller includes a central processing unit, memory and other functional elements, all on a single semiconductor substrate, or integrated circuit." (Ex. A, 2:2-5.) It also presents five different embodiments or applications of a microcontroller: (1) in a smartcard, which includes a microcontroller that is mounted to a plastic card (*id.*, Fig. 21); (2) mounted in a module for telephone or mobile phone (*id.*, Fig. 22); (3) in a key ring (*id.*, Fig. 23); (4) in a ring (*id.*, Fig. 24); and (5) in an electrical subsystem of an automobile (*id.*, Fig. 25). All of these are illustrations of exemplary embedded applications in that the microcontroller is embedded within a device and designed to be used for a specific

[6] This term is not a limitation in claim 7 of the '485 Patent and its dependents. (Ex. L, '485 Patent, 19:59 - 20:6.)

purpose or set of purposes in contrast to a general purpose computer. Contemporaneous extrinsic evidence is consistent: "another term to describe a microcontroller is an *embedded* controller."[7]

In contrast with Gemalto's straightforward construction, Defendants impermissibly seek to read in two negative limitations. First they state that a microcontroller "does not require external components to function properly." This is based on a clever distortion of the specification: "[m]icrocontrollers differ from microprocessors in many ways. For example, a microprocessor *typically* has a central processing unit that requires *certain* external components (e.g., memory, input controls and output controls) to function properly." (*Id.*, 1:63-66). By merely stating that a microprocessor *typically* requires *certain* external components to function properly, it does not necessarily follow that a microcontroller does not need *any* external components to function properly as Defendants suggest. In fact, the specification affirmatively states a microcontroller *requires* external components: "[d]ue to the small number of *external components required* and their small size, microcontrollers …" (Ex. A, 2:35-36.) The extrinsic evidence is in accord, demonstrating that microcontrollers typically use external components to function properly. [8]

[7] Ex. M, The Microcontroller Idea Book, p. 1, 3 (microcontrollers are "small computers dedicated to one task or a set of closely related tasks" with on-chip memory and I/O interfaces in addition to the CPU, which is also found in microprocessors.)

[8] Ex. N, U.S. Patent No. 5,796,312 ("*[m]ost microcontrollers use an external clock source* because they are inherently accurate, as compared to providing an internal oscillator, which is inherently inaccurate for use by the microcontroller."); Ex. O, U.S. Patent No.5,809,544 ("*typical microcontrollers have* what is known as an expansion bus, that is a bus with address and data lines that allows *external memory* or memory-mapped peripherals to be accessed."); Ex. P, U.S. Patent No. 6,260,101 ("[a] *microcontroller is typically coupled to one or more external memory* devices which store software programs consisting of instructions and data. During operation, the microcontroller fetches the instructions and data from the external memory devices and operates upon the data during instruction execution.")

Defendants misconstrue the intrinsic evidence in introducing the negative limitation stating that a microcontroller is "not a microprocessor." As stated in the specification "microprocessor implementations frequently are used in desktop and personal computers." As explained above, a microcontroller is a microprocessor designed for a specific purpose and includes memory and other functional elements together on a single substrate or integrated circuit. In other words, rather than suggesting that the two are opposites, the intrinsic evidence shows that a microcontroller is a special type of microprocessor. The specification of the provisional application states that "[o]ne of the uses of microcontrollers is for integrated circuit cards" (Ex. Q, Provisional at 1:28.) It then states that "[i]ntegrated circuit cards are also known in the art as chip cards, *microprocessor cards,* or smartcards." (*Id.*, 2:18-19.) When discussing the card and its operating system it states: [t]he low level OS manages the different resources of the *microprocessor* component." Figure 3 also shows a block diagram with "*Microprocessor* Hardware on the Smartcard." (*Id.*, 37.) All of these references to microprocessors and microcontroller indicate that a microcontroller is a type of microprocessor. The remaining statements relating to microprocessor use qualifying language such as "typical," "typically" or "frequently." (*See, e.g.*, Ex. A, 2:5-35.) Such qualified language falls far short of the unequivocal language required to disclaim all microprocessors. *Playtex Prods.*, 400 F.3d at 908. Therefore the "not a microprocessor" limitation is not supported by the intrinsic evidence and would only provide Defendants a backdoor to associate additional limitations not cited within the intrinsic record.

2) Integrated Circuit Card

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
integrated circuit card	an integrated circuit containing a central processing unit, memory and other functional elements on a base	a card containing a central processing unit, memory and other functional elements, all on a single semiconductor substrate or integrated circuit, that does not require external components to function properly; not a microprocessor system

Plaintiff contends that an integrated circuit card is an "an integrated circuit containing a central processing unit, memory and other functional elements on a base" while Defendants seek to read in the same three negative limitations presented above for "microcontroller.".

The specification states that "[t]he integrated circuit card includes a *processor* [central processing unit] that is coupled to the *memory* and is configured to use the interpreter to execute the first instructions and to communicate with the terminal via the *communicator* [other functional elements]." (Ex. A, 5:36-40.) It also does not limit the processor to a microcontroller: "[t]he processor may be a microcontroller." (*Id.*, 4:12-13.) The elements of an integrated circuit card do not have to be mounted on a plastic card or in the case of a microcontroller a single semiconductor substrate: "[i]n other embodiments, … *mounted within bases* other than a plastic card." (*Id.*, 7:59-62.) For these reasons and others presented above in connection with the term "microcontroller," the negative limitations presented by defendants are improper.

3) **Programmable Device**[9]

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
programmable device	No construction necessary. To the extent that the term is construed, it should be given its plain and ordinary meaning: a device that can execute a computer program	integrated circuit card or other device controlled by a microcontroller; not a microprocessor based device or system

Plaintiff believes that the term "programmable device" does not need construction, but to the extent this term is construed it should be given its plain and ordinary meaning: "a device that can execute a computer program" whereas Defendants seek to limit this term only to devices controlled by microcontrollers. First, Defendants' construction lacks merit because of the direct intrinsic evidence and other reasons discussed above with respect to the term "microcontroller." Second, even without such direct evidence a patentee may claim a genus of the species microcontroller in the electrical arts given the predictability in this field. *Cedarapids, Inc. ex rel.*

[9] This term is not a limitation in claim 3 of the '727 Patent and its dependents. (Ex. R, '727 Patent, 19:29-43.)

El-Jay Div. v. Nordberg, Inc., No. 95-1529, 1997 U.S. App. LEXIS 21157 (Fed. Cir. Aug. 11, 1997) ("In cases involving predictable factors, such as mechanical or electrical elements, a single embodiment provides broad enablement in the sense that, once imagined, other embodiments can be made without difficulty and their performance characteristics predicted by resort to known scientific laws."). The invention disclosed relates to conversion of byte codes and the invention is applicable to other programmable devices, not just the preferred embodiment.

d. Resource Constraints

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
resource constraints	computing resources that are limited when compared to conventional computing platforms, such as microprocessor-based desktop and personal computers	Indefinite

The next set of terms relate to "resource constraints." Gemalto respectfully submits a construction based on the intrinsic evidence as it would be understood by a person of ordinary skill in the art—"computing resources that are limited when compared to conventional computing platforms, such as microprocessor-based desktop and personal computers." Rather than offer a construction for a term widely used and understood even in the context of their own patents, Defendants offer no construction.

Gemalto's construction is clear and is based on how the resource constraints are described in the specification. Recognizing that all devices have a finite amount of resources at their disposal, the patentees contrasted those platforms that had sufficient resources to run Java-based programs and those that do not. When it was introduced in 1995, Java was targeted and developed for use in workstations and personal computers (*i.e.*, general purpose computers), or as the patents refer to them, "conventional platforms" and "microprocessor implementations frequently … used in desktop and personal computers." (Ex. A, 1:55-59.) Noting that there are no corresponding Java implementations for microcontrollers, such as "would typically be used in a smartcard" (*id.*, 1:59-61), the patentees set up the distinction between the baseline resources of one class of devices (microprocessor-based desktop and personal computers, i.e., general purpose computing platforms) and the reduced resources of another class of devices (microcontroller-based devices, i.e. computing platforms for embedded applications or specific purposes). (*See Id.*, 1:52-54 (a Java application and Java virtual machine "must be written [such] that [it] will run within the *constraints* of the platform… [and] a mechanism must be provided for loading the desired Java application on the platform, again keeping within the *constraints* of the platform.").) The microcontroller-based implementations of the Gemalto patents are "resource constrained"—they have computing resources that are limited when compared to conventional computing platforms, such as microprocessor-based desktop and personal computers.

The patents expand on the difference between the two classes of devices. "Conventional platforms that support Java are typically *microprocessor-based computers*, with access to relatively large amounts of memory and hard disk storage. Such microprocessors implementations frequently are used in *desktop and personal computers*." (*Id.*, 1:57-59.) Memory is one example of a resource that may be constrained on some devices such as a microcontroller-based smartcard, "[a]s compared to the relatively large external memory accessed by the microprocessor," in a typical desktop computer running Java. (*Id.*, 2:5-16.) The reason is that "[i]n a microcontroller, the amount of each kind of memory [(RAM, ROM and EEPROM)] available is *constrained* by the amount of space on the integrated circuit used for each kind of memory." (*Id.*, 2:14-16.) The specification also explains that the Card JVM may encounter a resource limitation when it is executing. (*Id.*, 15:37-39.) Again, the specification makes it clear that "resource constraints" are "computing resources that are limited when compared to conventional computing platforms, such as microprocessor-based desktop and personal computers."

There is also ample extrinsic evidence, including from the employees of one of the Defendants, that those skilled in the art view the terms "resource constraints" and "resource constrained" as defining the relative differences in processing capability between two classes of devices:

- *Resource-constrained devices are generally considered to be those that are relatively restricted in memory and/or computing power or speed, as compared to typical desktop computers* and the like. By way of example, other resource constrained devices include cellular telephones … and other miniature or small footprint devices. (Ex. S, U.S.Patent No. 6,687,898, 3:1-8.)

- Mobile embedded devices in use today provide computing power, but *operate under increased resource constraints compared to their desktop or server counterparts*. While desktop computer systems often use microprocessors running at clock frequencies of 1 gigahertz or more, and are equipped with 256 MB of RAM or more, a typical handheld computer device has only a fraction of such power. (Ex. T, U.S. Patent No. 6,990,662, 1:54-64.)

- Typically the devices connected to the MICE comprise machines *such as personal computers* that have the necessary resources and/or capabilities to receive all of the feeds sent to a multicast group to which the machine is a member. However, to extend the functionality of the system, *it may be desirable to introduce a device having more limited resources and/or capabilities (also referred to herein as a resource-constrained device or RCD). The limited resources and/or capabilities may result from, for instance, the RCD having reduced processing speed, available bandwidth, etc.* or the RCD being limited to receiving one kind of data (e.g., only audio data). (Ex. U, U.S.Patent No. 7,801,068, 1:36-47 (assigned to Motorola).)

In addition to applying the same definition as Gemalto in the specification, these patents[10] also make use of "resource-constrained" in the claims. Defendants' indefiniteness argument rings hollow.

Moreover, the fact that this term is defined relative to its environment does not render the term indefinite. In *Orthokinetics, Inc. v. Safety Travel Chairs, Inc.*, 806 F.2d 1565, 1575-76

[10] There are numerous other U.S. patents that also use the term "resource constraints" in both the specification and in the claims. While too voluminous to include as exhibits hereto, (Appendix 2) includes a sample list of these patents and Gemalto will provide PDF copies of these patents on CD with the Court's hard copy of the brief and exhibits.

(Fed. Cir. 1986), for example, the Federal Circuit held that the term "front leg portion . . . so dimensioned as to be insertable through the space between the doorframe of an automobile and one of the seats thereof" was not indefinite, even though the scope of this claim, and therefore infringement, depended on the ultimate environment. Likewise, in *Intellectual Prop. Dev., Inc. v. UA-Columbia Cablevision of West*, 336 F.3d 1308 (Fed. Cir. 2003), the Court found the term "common optical fibre" not indefinite because "those skilled in the art would understand the scope of the claim when the claim is read in light of the specification." Finally, in *Source Search Techs., LLC v. LendingTree, LLC*, 588 F.3d 1063, 1076 (Fed. Cir. 2009) the Federal Circuit held that the use of the phrase "standard goods and services" not indefinite, even though what was "standard" would vary from situation to situation.

Defendants' reliance upon *PC Connector Solutions LLC v. SmartDisk Corp.*, 406 F.3d 1359 (Fed. Cir. 2005), is misplaced. Contrary to the implication in Defendants' letter brief, the Court in that case did not declare the term indefinite. Instead, the Court merely affirmed the district court's construction, which limited the claimed "conventional," "traditional," and "standard" I/O ports to "those in existence at the time of the filing in 1988." *Id*. at 1362. In rejecting the patentee's broader definition, the court merely noted that "[a] claim cannot have different meanings at different times." *Id*. at 1363. Gemalto's construction does not run afoul of this dicta because its construction does not vary. *Source Search Techs.*, 588 F.3d at 1076. Because Gemalto's construction is "sufficiently precise to permit a potential competitor to determine whether or not he is infringing, *Exxon Research & Eng'g Co. v. United States*, 265 F.3d 1371, 1376 (quoting *Morton Int'l, Inc. v. Cardinal Chem. Co.*, 5 F.3d 1464, 1470, (Fed. Cir. 1993)), the term is definite.

e. Means Plus Function Elements

Several of the asserted claims include certain "means plus function" elements, which the parties agree are governed by §112, ¶6. The only dispute is whether the inventors described the corresponding structure. The parties seem to agree that the basic structure corresponding to each element is a programmed computer. Where the parties diverge, however, is whether the inventors adequately described an "algorithm" to implement the recited function. Gemalto contends that the inventors provided step-by-step procedure for implementing the recited functions, in both flowchart and prose form, and thus satisfied its obligations under §112, ¶2. *See Typhoon Touch Techs., Inc. v. Dell, Inc*., 659 F.3d 1376, 1384 (Fed. Cir. 2011). Defendants, on the other hand, contend that the description is so lacking that the claims cover all computer-implemented means for performing the function, but cannot carry their heavy burden to invalidate these claims in light of the detailed description of these elements provided in the specification.

1) **"means for mapping"**[11]

The recited function is "mapping such strings to unique identifiers." "Such strings" refers to the earlier recitation of the "identifying strings for objects, classes, fields or methods" in the claims in which the "means for mapping" appears (e.g., Ex. A, at 25:24-27). The function of mapping strings is described as part of the "class file converter." (*Id.*, 9:19-23, Fig. 5.) The "class file converter" is part of the "terminal 14" that "prepares and downloads Java applications." (*Id.*, 7:66-67.) The "terminal" is also described as a personal computer, providing Java development support. (*Id.*, 7:51-55.) This computer forms the foundation for the structure for the means-plus-function limitations of the Gemalto Patents, including the "means for including attributes" limitation upon which the parties have agreed. (D.I. 157, JCCS, at 2.)

[11] Due to the length of the proposed constructions for the means-plus-function elements, the proposed constructions are set forth in Appendix 3, pp. 1-4.

The "mapping" aspect of the "class file converter" is shown in element 51b of Figure 5 and further described in detail at column 9. In essence, the mapping function is performed by replacing strings in the constant pool of the class file with a corresponding integer:

> This compaction [of the constant pool] is achieved by <u>mapping all the strings found in the class file constant pool 42 into integers</u> (the size of which is microcontroller architecture dependent). These integers are also referred to as IDs. Each ID uniquely identifies a particular object, class, field or method in the application 20. Therefore, *the card class file converter 26 replaces the strings in the Java class file constant pool 42 with its corresponding unique ID.*

(Ex. A, 9:29-37, Fig. 5.) By replacing strings in the constant pool with an integer, the converter significantly reduces the amount of memory required to store the constant pool. (*Id.*, 9:27-29.)

"Precedent and practice permit a patentee to express that procedural algorithm 'in any understandable terms including as a mathematical formula, in prose, or as a flow chart, or in any other manner that provides sufficient structure.'" *Typhoon Touch Techs., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1384 (Fed. Cir. 2011). (quoting *Finisar Corp. v. DirecTV Grp., Inc.*, 523 F.3d 1323, 1340 (Fed. Cir. 2008)). The Gemalto Patents provide a sufficiently detailed description "in prose" of the algorithmic structure required to implement the recited "mapping" function. *See Levine v. Samsung Telcoms. Am., LLC*, 2012 U.S. Dist. LEXIS 13528, at *56 (E.D. Tex.); *Typhoon Touch Techs.*, 659 F.3d at 1386. Defendants' argument to the contrary would require the patentee to incorporate the actual source code in order to provide a sufficient disclosure of the algorithm, but the law is clear that source code is not required to satisfy §112(2). *See, e.g, Id.*, at 1385-86 ("For computer-implemented procedures, the computer code is not required to be included in the patent specification.") Accordingly, this element should be construed to include a computer programmed to replace strings in the class file constant pool with fixed-size integers to compact the constant pool, is set forth in Figure 5 (element 51b) and in the prose of columns 7 and 9 of the specification, as shown above.

2) "means for translating"

The parties agree that this element is a means-plus-function element, but fundamentally disagree over what constitutes the function. Gemalto contends that function consists of just the phrase "translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter" while the Defendants contend that the function also includes all of the steps, e.g., "b.1" thru "b.3" for claim 65 of the '317 patent. Defendants' position ignores the import of *WMS Gaming Inc. v. International Game Tech.*, 184 F.3d 1339 (Fed. Cir. 1999), upon which it purports to rely for its indefiniteness argument.

In *WMS Gaming*, the "district court determined that the structure disclosed in the specification to peform the claimed function was 'an algorithm executed by a computer.'" *Id.* at 1348. While the patent disclosed a particular algorithm for performing the claimed "assigning" function, the district court did not include that particular algorithm as part of the corresponding structure. On appeal, the Federal Circuit held that such as construction was "overly broad" because it included all computer-implemented means for performing the function. *Id*. The Court therefore limited the corresponding structure for the "means for assigning" to:

> a microprocessor programmed to perform the algorithm illustrated in Figure 6. In other words, the disclosed structure is a microprocessor programmed to assign a plurality of single numbers to stop positions such that: 1) the number of single numbers exceeds the number of stop positions; 2) each single number is assigned to only one stop position; 3) each stop position is assigned at least one single number; and 4) at least one stop position is assigned more than one single number.

Id. at 1349.

Subsequent decisions have gone on to clarify what level of specificity is required to disclose the algorithm. In keeping with the letter and spirit of *WMS Gaming*, the Federal Circuit has made clear that "[p]recedent and practice permit a patentee to express that procedural algorithm 'in any understandable terms including as a mathematical formula, in prose, or as a flow chart, or in any other manner that provides sufficient structure.'" *Typhoon Touch*, 659 F.3d at 1385 (citations omitted). It is equally clear that the algorithm does not need to be described down to the actual source code. *Id*. at 1385-86 ("For computer-implemented procedures, the computer code is not required to be included in the patent specification.")

In the case of the "means for translating," the patentee added the algorithmic structure to the claim itself. The function is "translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter." The claim signals an end of the function and the beginning of the algorithmic structure with the use of the word "by." The algorithm for performing this function is illustrated in the flow chart of Figure 6 which, as shown in Appendix 3, pg. 5, closely follows the particular algorithm steps—"b.1" thru "b.3"—of the claims. Thus, the patent describes and claims a "step-by-step procedure for accomplishing [the translation] result." *Id*. at 1384-85 (quoting *In re Freeman*, 573 F.2d 1237, 1245 (CCPA 1978)). Because the algorithm is part of the claim, the element does not implicate the breadth concerns of *WMS Gaming* and its progeny such as *Aristocrat Techs. v. Int'l Game Tech.*, 521 F.3d 1328, 1331 (Fed. Cir. 2008). Accordingly, the Court should hold that the corresponding structure is a computer programmed to perform the algorithm/steps explicitly set forth in the claims.

The fundamental flaw in Defendants' indefiniteness argument is that it necessarily treats the algorithm steps (*e.g.*, "b.1" - "b.3") as part of the function, rather than part of the structure. This misapplies *WMS Gaming*. As described above, the algorithmic steps "1" thru "3" were part of the structure and not the function. The fact that the steps were not explicitly part of the claim is of no moment because they were read into the claim by operation of law. Had the patentee in *WMS Gaming* explicitly recited those steps as part of the claim, as patentee has done in this case, no more disclosure in the specification would have been necessary to satisfy §112(2).

Even assuming for the sake of argument that the algorithmic "steps" recited in the claim limitations are not part of the corresponding structure, the remainder of the specification provides even further description of how those steps, as shown in the chart set forth in Appendix 3, pg.6. Again, this "prose" description of how to implement each of these steps is more than sufficient to satisfy the definiteness requirement. *See Levine v. Samsung Telcoms. Am., LLC*, 2012 U.S. Dist. LEXIS 13528, at *56 (E.D. Tex.) (holding that the element is not invalid where the specification's "disclosures 'in prose'" of an algorithm was sufficient under §112").. The only remaining level of description is the actual source code itself. While the patentee provided source code in "Appendix D … incorporated by reference," ('317 patent, col. 10:28-29), the law is clear that source code is not required to satisfy §112(2). *Typhoon Touch Techs.*, 659 F.3d at 1385-86 ("For computer-implemented procedures, the computer code is not required to be included in the patent specification.") Accordingly, the Court should brush aside Defendants' indefininteness argument and adopt Gemalto's construction for this term.

f. **The Integrated Circuit Card**

Term	**Gemalto's Proposed Construction**	**Defendants' Proposed Construction**
"the integrated circuit card" in the '317 Patent, claims 58, 65, 78, 87, 88 and 92	"the microcontroller"	Indefinite

Due to an obvious and inadvertent mistake made by the patentees while amending the claims during reexamination of the '317 Patent, certain claims mistakenly recite "renumbering byte codes . . . in an instruction set supported by an interpreter on *the integrated circuit card*" (*see* Ex. A, 2:47-51 (emphasis added)), instead of reciting "renumbering byte codes . . . in an instruction set supported by an interpreter on *the microcontroller*." Because this correction is "not subject to reasonable debate" and the prosecution history confirms, rather than "suggest[s] a different interpretation of the claims," this Court should correct this obvious typographical error. *Novo Indus., L.P. v. Micro Molds Corp.*, 350 F.3d 1348, 1357 (Fed. Cir. 2003).[12]

[12] The patentees have also sought a certificate of correction from the PTO to correct this mistake. The patentees filed a petition with the PTO under 37 C.F.R. 1.181 for a certificate of correction, which was denied, and a petition for reversal of that decision, which was also denied. (Ex. V,

The ’317 Patent discloses and claims, among other things, executing applications written in a high level programming language on both an “integrated circuit card” and a “microcontroller.” The independent claims reflect this dichotomy: claims 1, 31, 64, 84-86, 93 and 94 are directed to an integrated circuit card (the “Integrated Circuit Card Claims”)) and claims 58, 65, 78, 87-88, and 91-92 are directed to a microcontroller (the “Microcontroller Claims”). (See Ex. A, Reexamination Certificate.) These classes of claims recite parallel claim limitations, except that one class recites “an integrated circuit card” in the preamble, while the other recites a “microcontroller” in the preamble. *Compare* claim 1 *with* claim 58. Both recite a “memory” and a “processor.” (*Id*.) Both also include a “derivative application” and an “interpreter” that “interprets” that application. (*Id*.) In both cases, the code is stored in the memory. In the case of the “integrated circuit card,” that memory is “on the integrated circuit card.” In the case of the “microcontroller,” however, both sides agree that the memory (and therefore the interpreter) is located “on the microcontroller.” (*Compare* D.I. 157, Exh. A at 6 with D.I. 157, Exh. B at 5.)

During reexamination, the patentee amended claim 1, directed to an integrated circuit card, by changing the limitation as follows:

> renumbering byte codes in [a compiled format] <u>the compiled form</u> to equivalent byte codes in <u>an instruction set supported by an interpreter on the integrated circuit card</u>[a format suitable for interpretation]; and

(Ex. X, ’317 Reexamination File History, 2/21/08 Reply and Amendment, at 2.) The patentee then made that same amendment to the “renumbering” limitation globally. (*Id*. at 3-24.) In the

’317 File History, 7/30/10 Petition for Correction.) On February 2, 2011, the patentees filed a petition with the PTO Director to reverse those denials, but that petition is currently still pending. (Ex. W, ’317 File History, 2/2/11 Petition for Correction.) Even though the PTO has not yet issued a certificate of correction, this Court may itself correct the mistake in the ‘317 Patent. *See Grantley Patent Holdings, Ltd. v. Clear Channel Communs., Inc.*, No. 9:06CV259, 2008 U.S. Dist. LEXIS 1588, at *48 (E.D. Tex. Jan. 8, 2008) (“[A] court may act to correct an error in the patent by interpreting that patent where no certificate of correction has been issued . . . .”).

accompanying remarks, the patentee made clear that all of the amendments "follow a similar pattern" to those in claim 1. (*Id*. at 30.) That is not surprising in light of the Examiner's rejection, which treated the claims together. (Ex. Y, '317 Reexamination File History, 12/21/07 Office Action, at 17 ("Per claim 58 … limitations of this claim are discussed above in claim 1.") The patentee also added new claims that propagated this error. (Ex. X at 25-29.) In doing so, the patentee accidentally neglected the fact that certain claims were directed to "a microcontroller," as opposed to an "integrated circuit card." Simply put, the patentee made a copy and paste error, pasting the amended limitation in the Integrated Circuit Card Claim globally, but neglecting to change "the integrated circuit card" to "the microcontroller" for the Microcontroller Claims.

The reference to "the integrated circuit card" in the Microcontroller Claims is clearly a mistake because there is no antecedent basis under 35 U.S.C. § 112, second paragraph, for "the integrated circuit card" in the Microcontroller Claims—"integrated circuit card" appears nowhere else in those claims. There is, however, an antecedent basis for "the microcontroller," since "microcontroller" appears in the preamble of the Microcontroller Claims, and those claims describe only microcontrollers and methods of programming microcontrollers. This would also make logical sense because the "interpreter" code is stored in the memory "on the microcontroller." The "Microcontroller Claims should therefore be corrected to read "an interpreter on *the microcontroller*." *See Novo Indus.*, 350 F.3d at 1357.

Both the specification and prosecution history of the '317 Patent are entirely consistent with the correction sought here. The Microcontroller Claims broadly refer to any type of device that includes a microcontroller and can use high level programming language — such as a mobile or fixed telephone, a key ring, jewelry, or an electrical subsystem of an automobile, as shown in the embodiments in Figures 22 through 25. (*See* Ex. A, 19:5-30.) The Integrated

Circuit Card Claims, on the other hand, narrowly refer to a specific type of device that includes a microcontroller and can use high level programming language — *i.e.*, an integrated circuit card, as shown in the embodiment in Figure 21. (*See id.*,18:65 –19:4.)

The prosecution history shows that the ’317 Patent was always prosecuted as including two classes of claims: a broad set directed to microcontrollers on devices generally, and a narrower set directed to microcontrollers on integrated circuit cards specifically. Thus, it is clear that the reference to “an interpreter on the integrated circuit card” in the Microcontroller Claims is a mistake, because for those claims the interpreter could be on a microcontroller that is on any number of devices, rather than limited to a microcontroller that is on an integrated circuit card. This Court should therefore correct the Microcontroller Claims to reference “an interpreter on the microcontroller.” *See Hoffer v. Microsoft Corp.*, 405 F.3d 1326, 1330-1331 (Fed. Cir. 2005) (“When a harmless error in a patent is not subject to reasonable debate, it can be corrected by the court, as for other legal documents. Here the error was apparent from the face of the patent, and that view is not contradicted by the prosecution history.”).

District courts readily correct typographical and cut and paste mistakes such as the one present in the Microcontroller Claims. For example, in *Interactive Tracking Systems v. Artafact LLC*, No. 08-10101-EFH, 2011 U.S. Dist. LEXIS 59270 (D. Mass. June 2, 2011), claim 1 of the patent-in-suit included the following limitations:

> (a) “a *respondent computer* interface for a respondent computer, said respondent computer interface for displaying, on a *respondent display* associated with each respondent computer;”
>
> (b) “a *client computer* interface for a client computer, said client interface for displaying, on a *client display* associated with each client computer;” and
>
> (c) “a *moderator computer* interface for a moderator computer, said moderator computer interface for displaying, on a *client display* associated with each moderator computer.”

(Ex. Z, U.S. Patent No. 6,256,663, 9:40 – 10:2.) The court corrected the phrase "client display" shown in limitation (c) to read "moderator display," concluding that "the inclusion of the word 'client' is an obvious typographical error," because claim 1 "set[] forth parallel descriptions of the client and respondent computers." *Interactive Tracking Sys.*, 2011 U.S. Dist. LEXIS 59270, at *8-9. That is, as shown above, limitation (a) referred to a "respondent computer" and a "respondent display," limitation (b) referred to a "client computer" and a "client display," and it was therefore clear that limitation (c), which referred to a "moderator computer," should be corrected to refer to a "moderator display" instead of a "client display." Similarly, in the '317 Patent the Integrated Circuit Card Claims refer to "the integrated circuit card," and it is clear that the parallel Microcontroller Claims should be corrected to refer to "the microcontroller" instead of "the integrated circuit card."[13]

g. **Access Control List**

Term	**Gemalto's Proposed Construction**	**Defendants' Proposed Construction**
access control list	a list which furnishes an indication of the types of access to be granted	list of elements of the form "type:identity:permissions" which furnishes an indication of types of access to be granted to a data file or communication channel.

Plaintiff proposes a construction for access control list ("ACL") that is consistent with its broad, plain and ordinary meaning while Defendants seek to limit it to the preferred embodiment. However, absent an express disclaimer by the patentee, it is improper to limit claim terms to preferred embodiments. *Playtex Prods.*, 400 F.3d at 908. The specification plainly states that the "access control list furnishes an indication of types of accesses to be granted to the identity, and based on the access control list, the processor selectively grants specific types of access . . .

[13] *See also LG Electronics Inc. v. Hitachi, Ltd.*, No. 5:07-CV-90, 2008 U.S. Dist. LEXIS 108377 at *100 (E.D. Tex. Dec. 8, 2008) (correcting recital of "as a result of *step(b)*" to instead read "as a result of *the step (a)*" since, as written, the recital was "self-referential, and thus nonsensical" and "the phrase 'as a result of the step (b)' . . . should clearly be a reference to step (a)."); *Grantley Patent Holdings, Ltd.*, 2008 U.S. Dist. LEXIS at *47-49 (E.D. Tex. Jan. 8, 2008) (correcting a dependent claim that referenced a "system" for accessing inventory information to instead refer to the "method" for accessing inventory information of the independent claims.).

to the requester." (Ex. A, 4:25-31.) In discussing the preferred embodiments, the specification states that an ACL is associated with every computational object and cites a data file or a communication channel as examples. (*Id.*, 16:36-42) The specification also describes a format arranged as "type:identity:permissions" under its preferred embodiments. (*Id.*) However, nowhere does patentee suggest that an ACL should be limited to this format or these types of computational objects and therefore it would be improper to adopt such a narrow interpretation.

h. **Terminal**

Term	Gemalto's Proposed Construction	Defendants' Proposed Construction
terminal	a device that communicates with the integrated circuit card or microcontroller	system, external to system controlling the ICC or microcontroller, that communicates with the ICC or microcontroller

Gemalto proposes that a terminal be construed as a "device that communicates with the integrated circuit card or microcontroller" while Defendants seek to add a limitation not supported by the specification. The patents recite: "[t]erminals can be automated teller machines (ATMs), point-of-sale terminals, door security systems, toll payment systems, access control systems, *or any other system that communicates with an integrated circuit card or microcontroller*." (Ex. A, 8:15-19) However, there is no requirement in the patents that the terminal be in a separate system than the microcontroller or integrated circuit card as Defendants suggest. For example, the same system can house both a terminal and a microcontroller. Further, Defendants additional limitation is awkwardly worded to suggest that the *system* housing a microcontroller *controls* the *microcontroller* when the opposite relationship is more natural. Accordingly, given that Patentee did not make an express disclaimer with respect to this term it would be improper to adopt Defendants' construction. *Playtex Prods.*, 400 F.3d at 908.

Dated: February 24, 2012

Respectfully submitted,

/s/ Sam Baxter
Sam Baxter
Texas State Bar No. 01938000
sbaxter@mckoolsmith.com
MCKOOL SMITH, P.C.
104 East Houston, Suite 300
Marshall, Texas 75670
Telephone: (903) 923-9000
Facsimile: (903) 923-9099

Robert A. Cote
rcote@mckoolsmith.com
Shahar Harel
sharel@mckoolsmith.com
Kevin Schubert
kschubert@mckoolsmith.com
MCKOOL SMITH, P.C.
One Bryant Park, 47th Floor
New York, New York 10036
Telephone: (212) 402-9400
Facsimile: (212) 402-9444

Peter J. Ayers
Texas State Bar No. 24009882
payers@mckoolsmith.com
Geoffrey L. Smith
Texas State Bar No. 24041939
gsmith@mckoolsmith.com
MCKOOL SMITH, P.C.
300 W. 6th St., Ste. 1700
Austin, Texas 78701
Telephone: (512) 692-8700
Facsimile: (512) 692-8744

ATTORNEYS FOR PLAINTIFF GEMALTO S.A.

CERTIFICATE OF SERVICE

The undersigned certifies that the foregoing document was filed electronically in compliance with Local Rule CV-5(a). As such, this document was served on all counsel who have consented to electronic services on this the 24th Day of February, 2012. Local Rule CV-5(a)(3)(A).

/s/ John Petrsoric
John Petrsoric